2. How Does SSRF Work?

How Does SSRF Work?

SSRF exploits the server's ability to make requests to other resources (internal or external). Here's a breakdown of the process:

---

1. The Setup: A Vulnerable Application

- The application includes a feature where it fetches or interacts with external resources based on user input.
  Examples:
  - A website that loads images from a URL provided by the user.
  - A server that retrieves data from an API endpoint specified in a user query.

---

2. The Attacker's Entry Point

- The attacker identifies an input field or parameter in the app that lets them control the **destination URL** of the server's request.
- Example:
  A form asks for an image URL:

```
GET /fetch?url=http://example.com/image.png
```

---

3. Crafting the Exploit

- The attacker replaces the `url` parameter with a **malicious URL**, redirecting the server to request something unintended.
  For example:
  - Accessing internal systems:

```
http://127.0.0.1/admin
```

  - Stealing metadata in cloud environments:

```
http://169.254.169.254/latest/meta-data
```

---

4. The Server Executes the Request

- The server makes the request, **trusting the attacker's input**.
- It might:

- Leak sensitive data from internal systems.
- Interact with internal APIs or services.
- Perform actions the attacker wouldn't normally have permission to do.

---

5. The Attacker Gets a Response

- The server's response reveals information or confirms that the malicious request was successful. Example:
  - The server fetches and returns admin data:

    ```
    Admin panel content: {"admin": "true"}
    ```

  - Or, it provides access to sensitive cloud information:

    ```
    IAM role: S3-read-only
    ```

Visual Example:

- **Legitimate Request**:

  ```
  GET /fetch?url=http://example.com/image.jpg
  Server fetches the image and returns it.
  ```

- **Malicious Request**:

  ```
  GET /fetch?url=http://127.0.0.1/admin
  Server fetches the admin panel data and leaks it to the attacker.
  ```

Real-World Scenarios Where SSRF Can Happen:

1. **Image Loaders:**
   Applications that retrieve images from user-supplied URLs.
2. **PDF Generators:**
   Fetch external files to include in documents.
3. **Webhook Handlers:**
   Process callback URLs for automation.
4. **API Gateway Proxies:**
   Forward requests to other services based on user input.